DATE

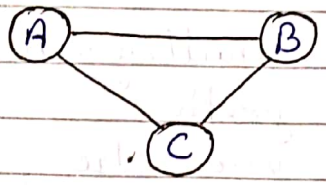
Graph

Graph is a group of vertices and edges

A graph G can be defined as an ordered set G(V, E) where V(G) represents the set of vertices and E(G) represents the set of edges which are used to connect these vertices.

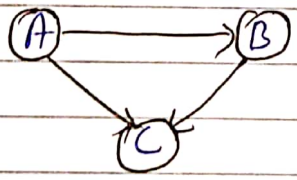
Un-directed graph

In an undirected graph, edges are not associated with the directions with them.



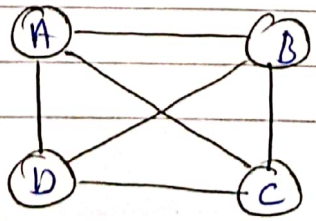
Directed graph

In a directed graph, edges are associated with the directions ~~with~~ with them, forming an ordered pair.



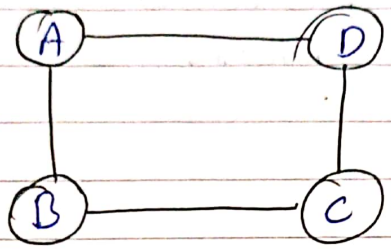
Complete graph

A graph is said to complete graph if all the nodes of the graph is connected to all other nodes



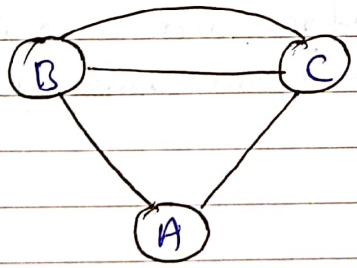
Cycle graph

In a cycle graph, vertices must be 3 or greater than 3 and all the vertices ~~of the each vertices~~ is of degree 2.



Multi graph

A graph is said to be multigraph, if parallel edges are present in the graph.
If there is more than one edge present between two vertices, then that pair of vertices is said to be having parallel edge



DATE

Breath First Search (BFS)

Step-1 : set status = 1 (ready state) for each node in G

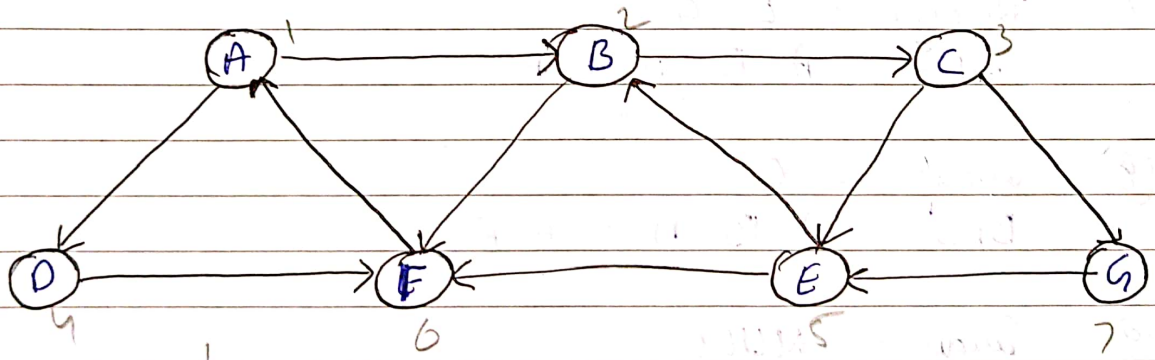
Step-2 : Enqueue the starting node A and set its status = 2 (waiting state)

Step-3 : Repeat step 4 and step 5 until Queue is empty.

Step-4 : Dequeue a node N. Process it and set its status to 3 (Processed state)

Step-5 : Enqueue all the neighbours of N that are in ready state and set their status = 2 (waiting state)

Step-6 : EXIT



1 A → 2 B, 4 D
2 B → 3 C, 6 F
3 C → 5 E, 7 G
4 D → 6 F
5 E → 2 B, 6 F
6 F → 1 A
7 G → 5 E

DATE

① Queue = NULL
BFS = NULL

② Queue = A
BFS = NULL

③ Queue = B D
BFS = A

④ Queue = B D C F
BFS = A B

⑤ Queue = C F
BFS = A B D

⑥ Queue = F E G
BFS = A B D C

⑦ Queue = E G
BFS = A B D C F

⑧ Queue = G
BFS = A B D C F E

⑨ Queue = NULL
BFS = A B D C F E G

Sequence : A(1) → B(2) → D(4) → C(3) → F(6) → E(5) → G(7)

DATE

Depth First Search (DFS)

Step-1: Set status = 1 (ready state) for each node in G

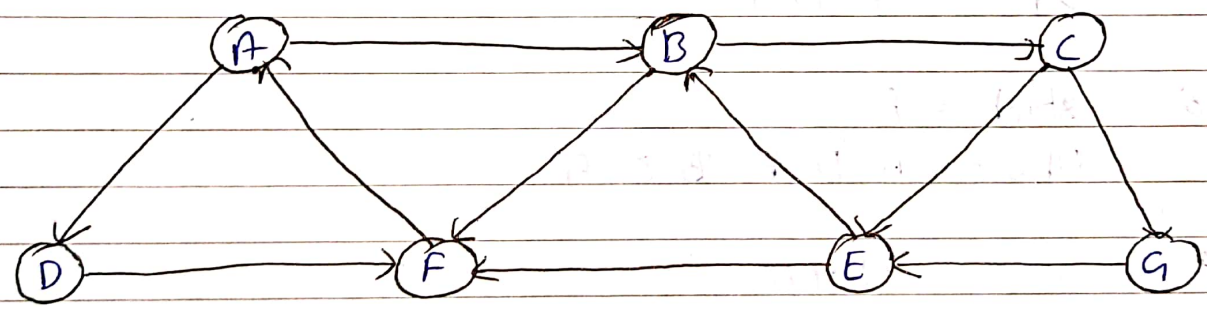
Step-2: Push the starting node A on the stack and set its status = 2 (waiting state)

Step-3: Repeat step 4 and step5 until stack is empty

Step-4: Pop the top node N. Process it and set its status = 3 (processed state)

Step 5: Push on the stack all the neighbours of N that are in ready state and set their status = 2 (waiting mode)

Step-6: EXIT



A → B, D
B → C, F
C → E, G
D → F
E → B, F
F → A
G → E

① Stack = NULL
DFS = NULL

② stack = A
DFS = NULL

③ stack = B D
DFS = A

④ stack = B F
~~stack~~ DFS = A D

⑤ stack = B
DFS = A D F

⑥ stack = C
DFS = A D F B

⑦ stack = E G
DFS = A D F B C

⑧ stack = E
DFS = A D F B C G

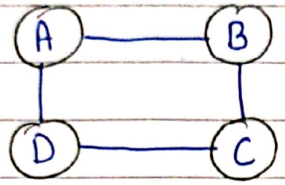
⑨ stack = NULL
DFS = A D F B C G E

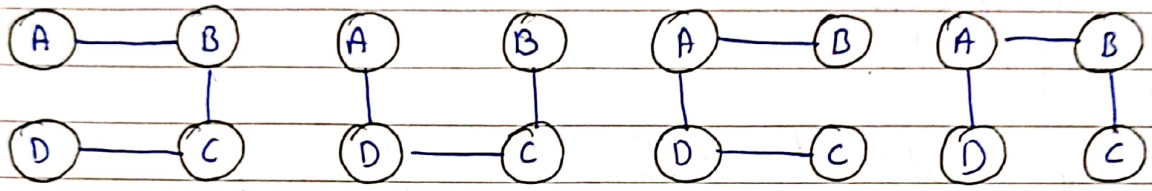
DATE

Spanning Tree

A spanning tree is a subset of graph G, which has all the vertices covered with minimum possible numbers of edges. Hence, a spanning tree does not have cycles and it can't be disconnected, for n vertices there will be (n-1) edges

eg:

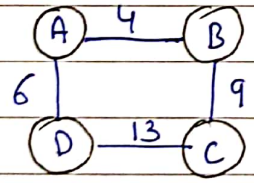


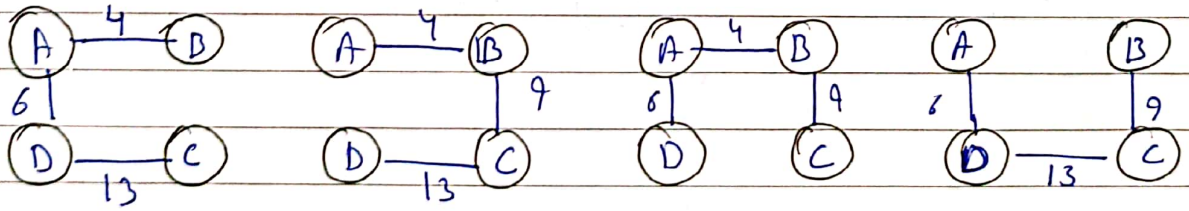


Minimum Spanning tree

In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight that all the spanning tree of the same graph

eg:





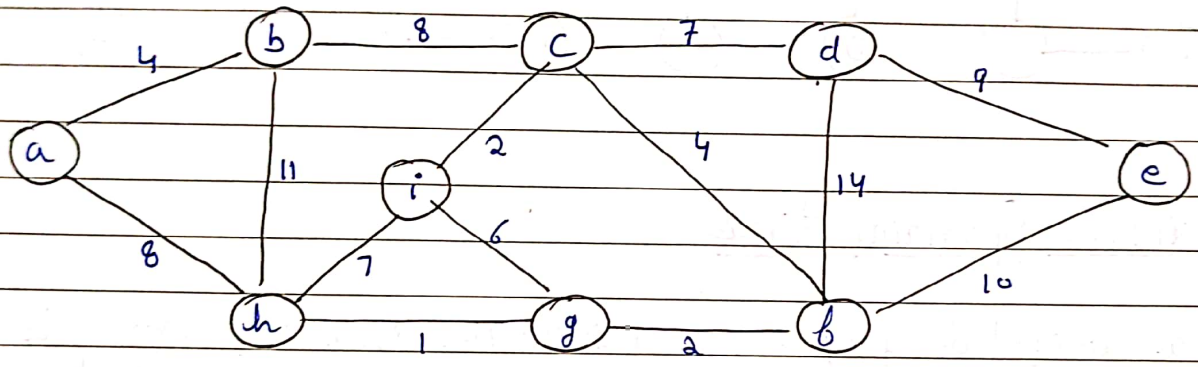
non-decreasing → increasing + equals to

DATE

Kruskal Algorithm Complexity = O(E log V)

1) A = ϕ
2) for each vertex v ∈ G.V
3) make set v
4) sort the edge G.E into non-decreasing order by weight
5) for each edge (u,v) ∈ G.V, taken in non-decreasing order by weight
6) if find-set (u) != find set (v)
7) A = {A ∪ (u,v)}
8) Union (u,v)
9) return A.

Example - 1



(a, b) = 4
(a, h) = 8
(b, h) = 11
(b, c) = 8
(c, i) = 2
(i, h) = 7
(h, g) = 1
(g, f) = 2
(c, f) = 4
(c, d) = 7
(d, f) = 14
(d, e) = 9
(e, f) = 10
(i, g) = 6

Step-1

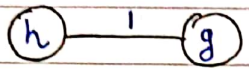
$V = \{a, b, c, d, e, f, g, h, i\}$

$A = \phi$

$(U,V) = \{(h,g), (g,f), (c,i), (g,f), (a,b), (c,f), (i,g), (i,h), (c,d), (a,h), (b,c), (d,e), (e,f), (b,h), (d,f)\}$

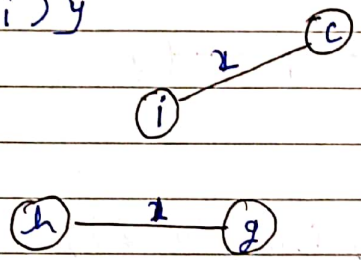
Step-2

$A = \{(h,g)\}$



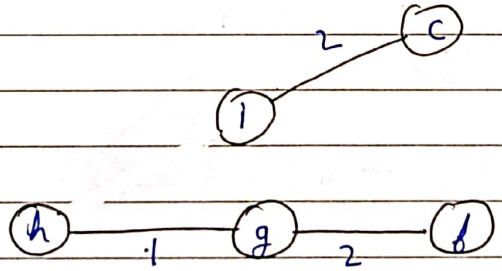
Step-3

$A = \{(h,g), (c,i)\}$

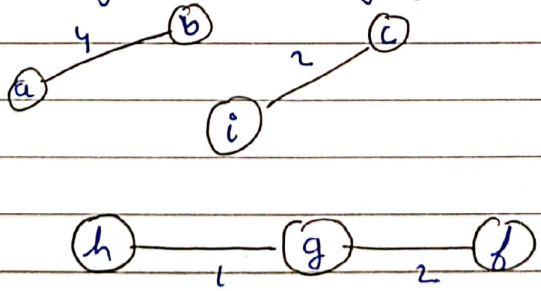


Step-4

$A = \{(h,g), (c,i), (g,f)\}$

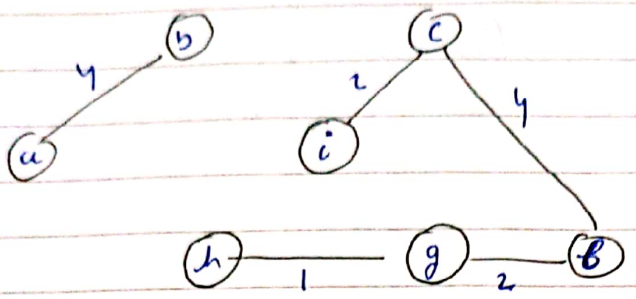


Step-5 $A = \{(h,g), (i,c), (g,f), (a,b)\}$

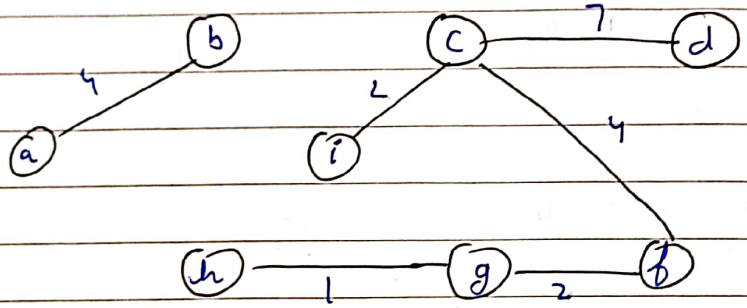


DATE

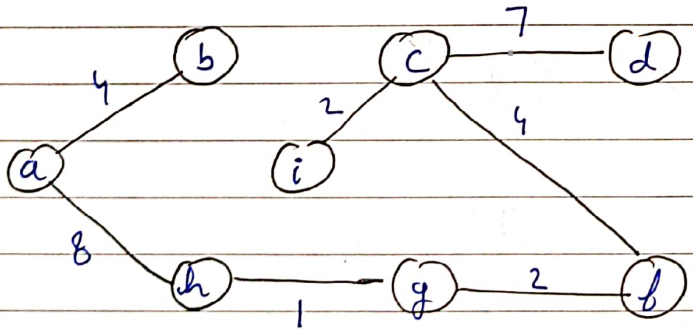
Step-6 $A = \{(h,g), (i,c), (g,f), (a,b), (c,f)\}$



Step-7 $A = \{(h,g), (i,c), (g,f), (a,b), (c,f), (c,d)\}$

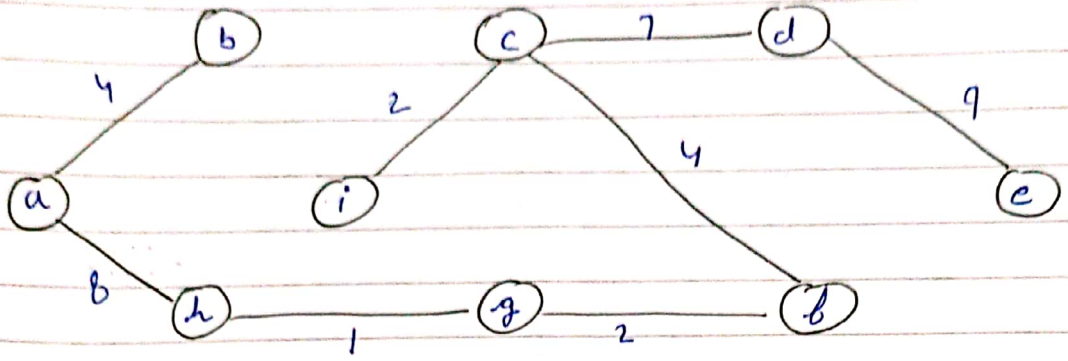


Step-8 $A = \{(h,g), (i,c), (g,f), (a,b), (c,f), (c,d), (a,h)\}$



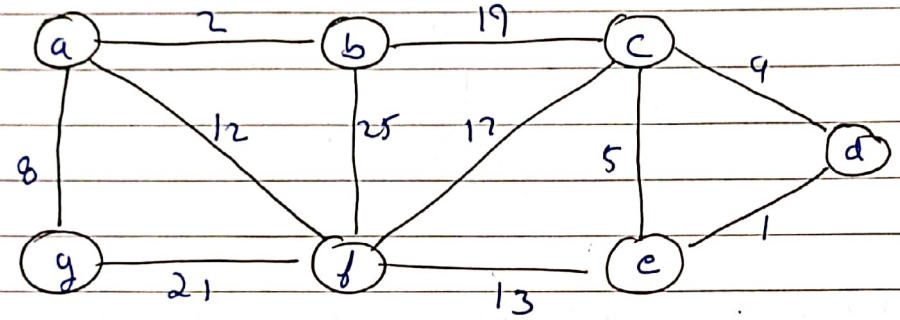
DATE

Step-1 $A = \{(h,g), (i,c), (g,f), (a,b), (c,f), (c,d), (a,h), (d,e)\}$



Minimum Weight = 37

Example-2



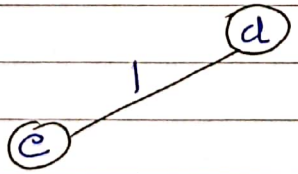
$(a,b) = 2$	$(b,f) = 25$	$(e,f) = 13$
$(b,c) = 19$	$(c,f) = 7$	$(f,g) = 21$
$(a,g) = 8$	$(c,d) = 9$	$(e,c) = 5$
$(a,f) = 12$	$(d,e) = 1$	

Step-1 $V = \{a, b, c, d, e, f, g, h\}$.

$A = \phi$

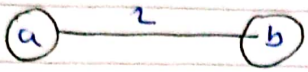
$(U,V) = \{(d,e), (a,b), (e,c), (a,g), (c,d), (a,f), (e,f), (b,c), (f,g), (b,f)\}$

Step-2 $A = \{(d,e)\}$

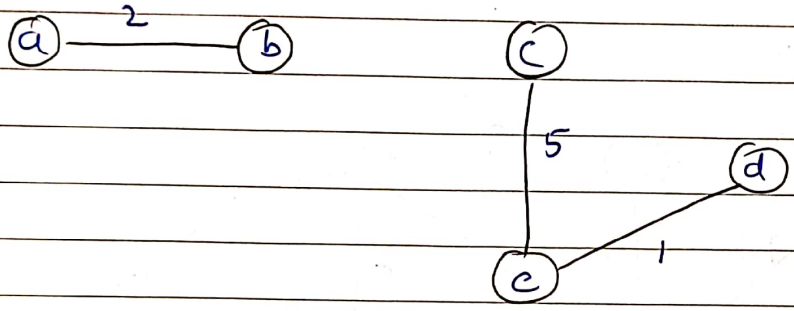


DATE

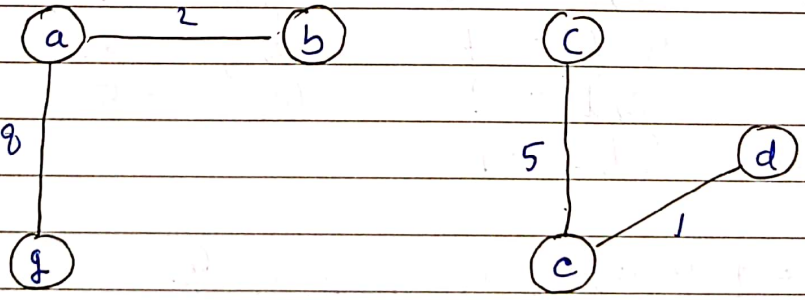
Step-2 $A = \{(d,e), (a,b)\}$



Step-3 $A = \{(d,e), (a,b), (e,c)\}$

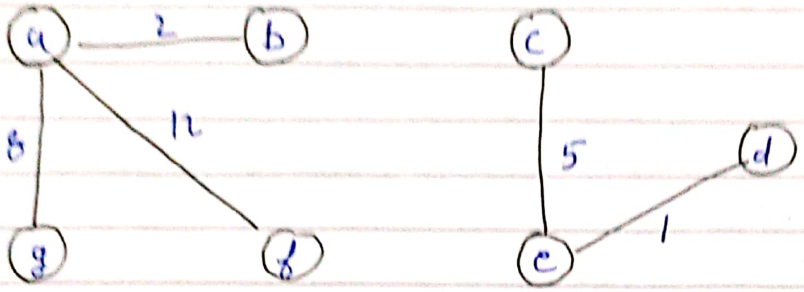


Step-4 $A = \{(d,e), (a,b), (e,c), (a,g)\}$

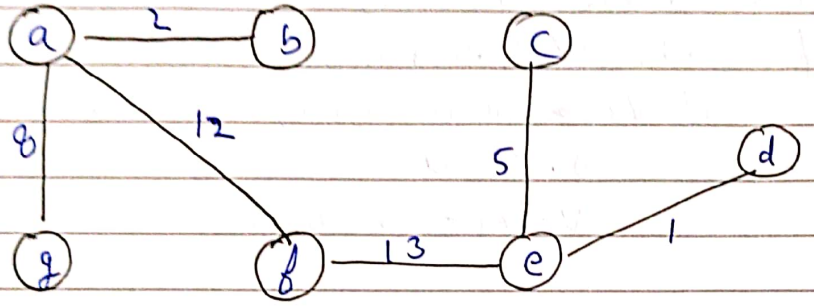


DATE

Step-5 A = {(d,e), (a,b), (e,c), (a,g), (a,f)}



Step-6 A = {(d,e), (a,b), (e,c), (a,g), (a,f), (f,e)}



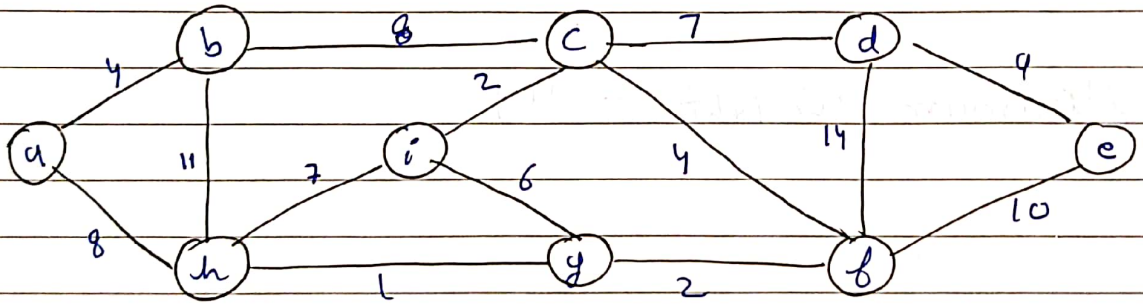
Minimum Weight = 41

DATE

Prim's Algorithm

complexity : $O[E \log V + V \log V]$
$O(E + V \log V)$ [if we use fibonacci heap]

```
for each u ∈ G.V
        u.key = ∞
        v.π = nil
                ↳ π is representing parent
r.key = 0
Q = G.V
while (Q ≠ 0)
        u = exctract-min(Q)        → Adjacent of u
        for each v ∈ G.[Adj[u]]
                if v ∈ Q and w(u,v) < v.key
                                      ↳ weight
                        v.π = u
                        v.key = w(u,v)
```



Step-1

Vertex	a	b	c	d	e	f	g	h	i
key value	0	∞	∞	∞	∞	∞	∞	∞	∞
Parent	nil	nil	nil	nil	nil	nil	nil	nil	nil

DATE

Step-2

Vertex	a	b	c	d	e	f	g	h	i
key value	0	4	∞	∞	∞	∞	∞	8	∞
Parent	nil	a	nil	nil	nil	nil	nil	a	nil

Step-3

Vertex	a	b	c	d	e	f	g	h	i
key value	0	4	8	∞	∞	∞	∞	8	∞
Parent	nil	a	b	nil	nil	nil	nil	a	nil

Step-4

Vertex	a	b	c	d	e	f	g	h	i
key value	0	4	8	∞	∞	∞	1	8	7
Parent	nil	a	b	nil	nil	nil	h	a	h

Step-5

Vertex	a	b	c	d	e	f	g	h	i
key value	0	4	8	∞	∞	2	1	8	6
Parent	nil	a	b	nil	nil	g	h	a	g

Step-6

Vertex	a	b	c	d	e	f	g	h	i
key value	0	4	~~8~~	14	10	2	1	8	6
Parent	nil	a	~~b~~	f	~~g~~	g	h	a	g

DATE

Step-7

Vertex	a	b	c	d	e	f	g	h	i
key value	0	4	4	7	10	2	1	8	2
Parent	nil	a	b	c	f	g	h	a	c

Step-8

Vertex	a	b	c	d	e	f	g	h	i
key value	0	4	4	7	10	2	1	8	2
Parent	nil	a	b	c	f	g	h	a	c

Step-9

Vertex	a	b	c	d	e	f	g	h	i
key value	0	4	4	7	9	2	1	8	2
Parent	nil	a	b	c	d	g	h	a	c

Minimum Weight = 37

DATE

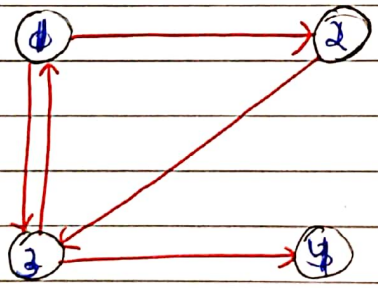
Transitive closure

It is the reach ability matrix to reach from vertex u to vertex v of a graph.
One graph is given, we have to find a vertex v which is reachable from another vertex u for all vertex pair (u, v)

Algorithm

1) Begin
2) Copy the Adjecency matrix into another matrix name D
3) ← for any vertex k in the graph, D
4) for ~~any~~ each vertex i in the graph, D
5) for each vertex j in the graph, D
6) D[i,j] = D[i,j] OR D[i,k] AND D[k,j]
7) EXIT
8) EXIT
9) EXIT
10) END

Example



D^0 = Adjecency matrix

	1	2	3	4
1	1	1	1	0
2	0	1	1	0
3	1	0	1	1
4	0	0	0	1

AND, OR 1

D^1 =

1	1	1	0
0	1	1	0
1	1	1	1
0	0	0	1

AND, OR 0

DATE

$D^2 =$

1	1	1	0
0	1	1	0
1	1	1	1
0	0	0	1

$D^3 =$

1	1	1	0
0	1	1	0
1	1	1	1
0	0	0	1

$D^3 =$

1	1	1	1
1	1	1	1
1	1	1	1
0	0	0	1

$D^4 =$

1	1	1	1
1	1	1	1
1	1	1	1
0	0	0	1

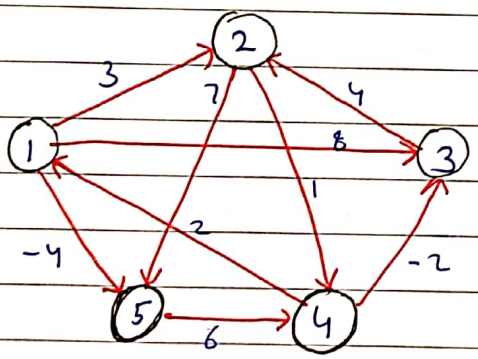
DATE

Floyd's Warshall Algorithm

Algorithm

1) n = row of W
2) $D^0 = W$
3) for K = 1 to n
4) for i = 1 to n
5) for j = 1 to n
6) do $d_{ij}^{(k)} = \min \{ d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \}$
7) EXIT
8) EXIT
9) EXIT
10) return D

Example



$$D_0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix} \end{array}$$

DATE

$$D^1 = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

$$D^2 = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$$

$$D^3 = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 0 & \infty & \infty & 6 & 0 \end{bmatrix}$$

$$D^4 = \begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

$$D^5 = \begin{bmatrix} 0 & 1 & -3 & -2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$$

DATE

Dijkstra Algorithm [single source shortest path]

Initialize single source (G, S)

1. S = ϕ
2. Q = G.V
3. while Q ≠ ϕ
4. u = exhact-min (Q)
5. S = S ∪ {u}
6. for each vertex v ∈ G.Adj [u]
7. RELAX (u, v, w)

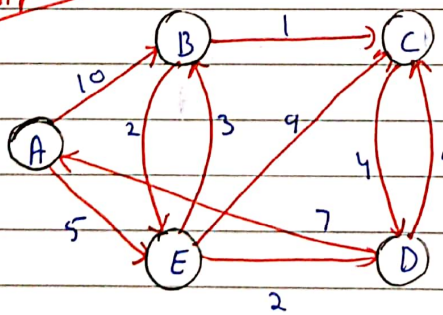
Initialize single source (G, S)

1. for each vertex v ∈ G.V
2. v.d = ∞
3. v.π = nil
4. S.d = 0

RELAX (u, v, w)

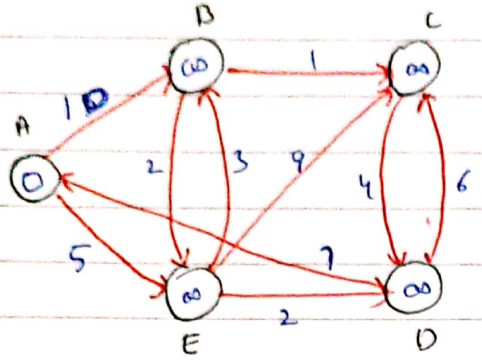
1. if v.d > ud + w(u, v)
2. v.d = u.d + w(u, v)
3. v.π = u

Example-1



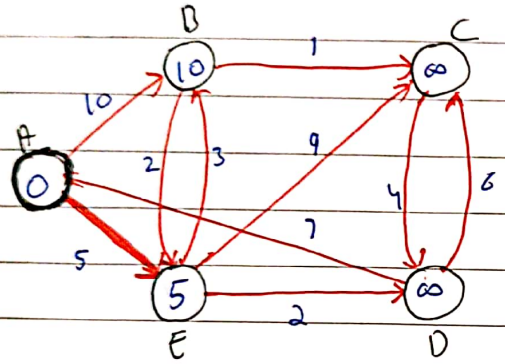
DATE

Step - 1



A	B	C	D	E
0	∞	∞	∞	∞
	10	∞	∞	5
	8	14	7	

Step - 2



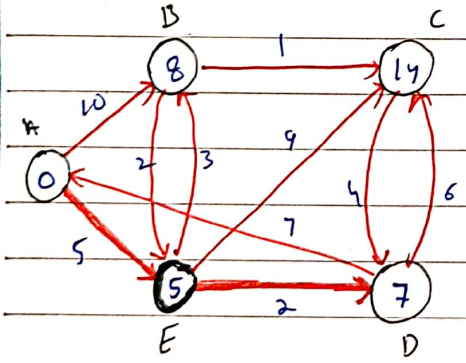
node B : $\infty > 0+10$

V.d = 0+10 = 10

node E : $\infty > 0+5$

V.d = 0+5 = 5

Step - 3



node B: $10 > 5+3$

V.d = 5+3 = 8

node C : $\infty > 5+9$

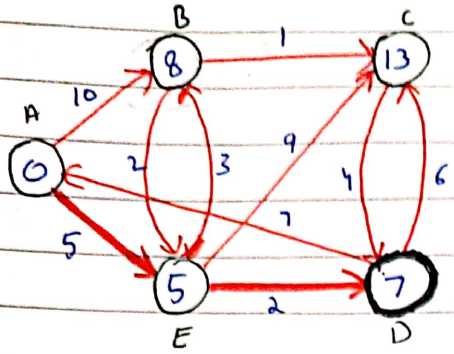
V.d = 5+9 = 14

node D: $\infty > 2+5$

V.d = 7

DATE

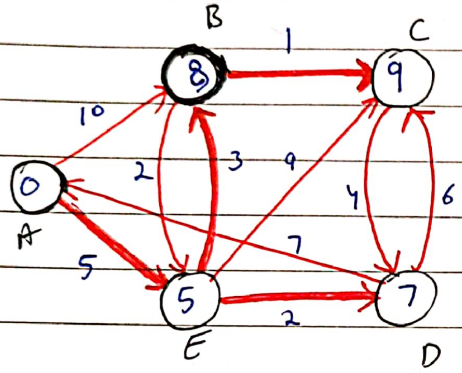
Step-4



node C : 14 > 7+6

V.d = 6+7 = 13

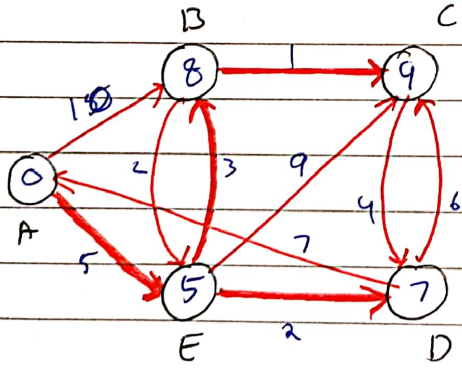
Step-5



node C : 13 > 8+1

V.d = 8+1 = 9

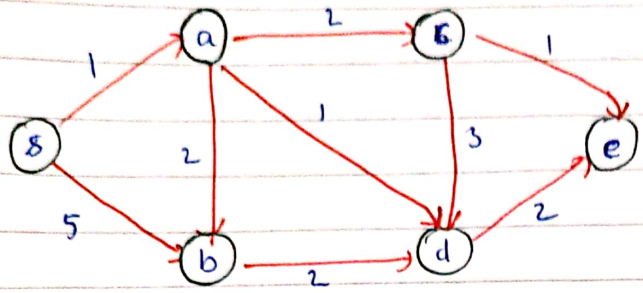
Final Graph



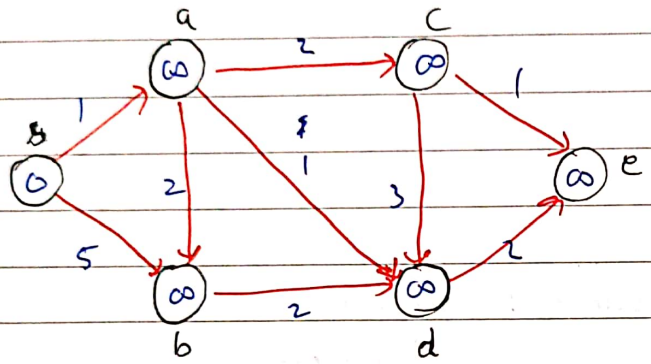
weight = 11

DATE

Example-2



Step-1



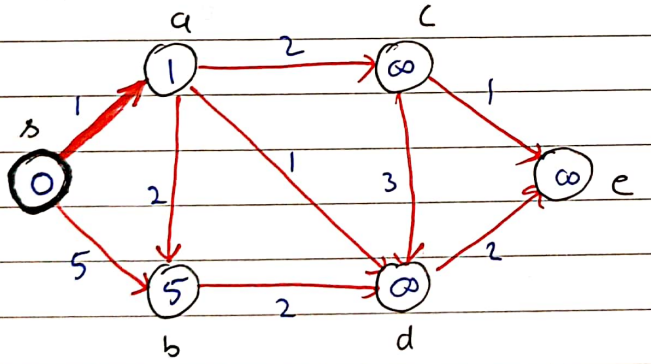
node a : $\infty > 0+1$

v.d = 1

node b : $\infty > 0+5$

v.d = 5

Step-2



node c : $\infty > 1+2$

v.d = 3

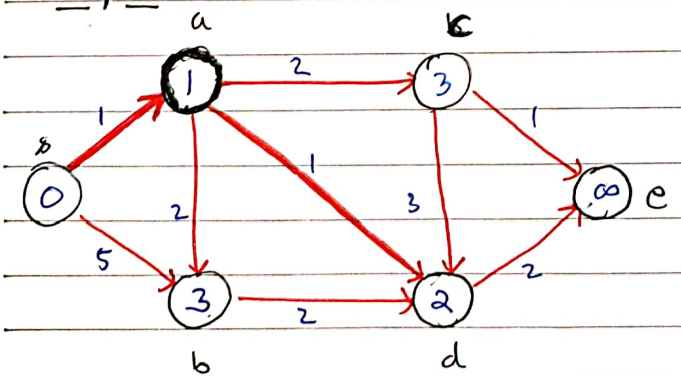
node d : $\infty > 1+1$

v.d = 2

node b : $5 > 2+1$

v.d = 3

Step-3

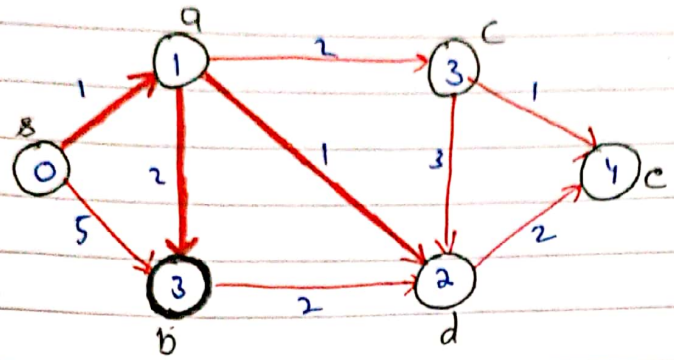


node e : $\infty > 2+2$

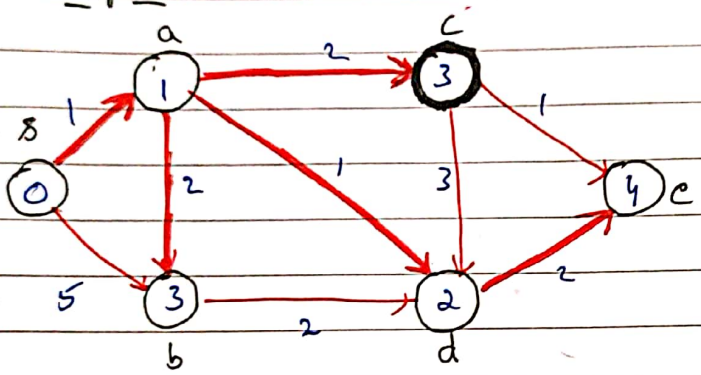
v.d = 4

DATE

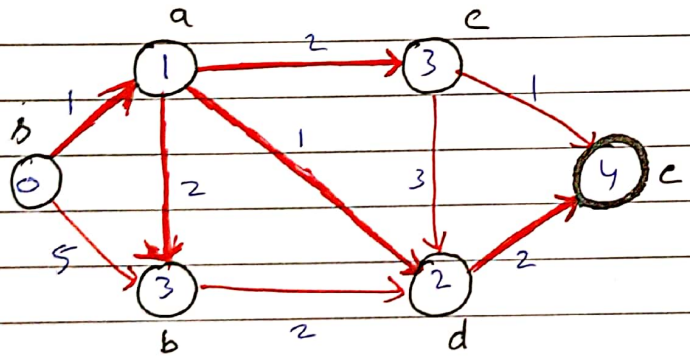
Step-4



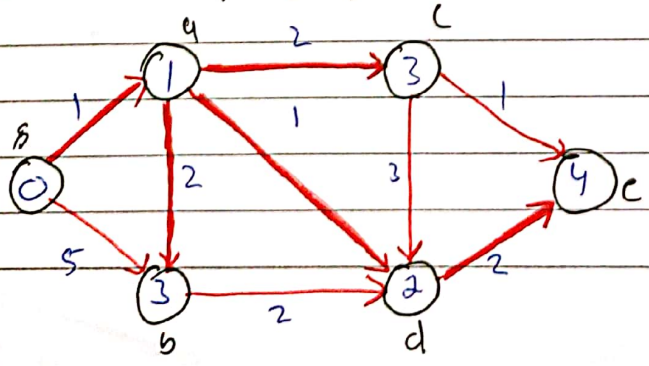
Step-5



Step-6



Final ~~Step~~ Graph



weight = 8